Auton Agent Multi-Agent Syst (2017) 31:1–35
DOI 10.1007/s10458-015-9311-7

Mobile crowdsensing with mobile agents

Teemu Leppänen[1] · **José Álvarez Lacasia**[2] · **Yoshito Tobe**[3] · **Kaoru Sezaki**[2] · **Jukka Riekki**[1]

Published online: 24 October 2015

Abstract We introduce mobile agents for mobile crowdsensing. Crowdsensing campaigns are designed through different roles that are implemented as mobile agents. The role-based tasks of mobile agents include collecting data, analyzing data and sharing data in the campaign. Mobile agents execute and control the campaign autonomously as a multi-agent system and migrate in the opportunistic network of participants' devices. Mobile agents take into account the available resources in the devices and match participants' privacy requirements to the campaign requirements. Sharing of task results in real-time facilitates cooperation towards the campaign goal while maintaining a selected global measure, such as energy efficiency. We discuss current challenges in crowdsensing and propose mobile agent based solutions for campaign execution and monitoring, addressing data collection and participant-related issues. We present a software framework for mobile agents-based crowdsensing that is seamlessly integrated into the Web. A set of simulations are conducted to compare mobile agent-based campaigns with existing crowdsensing approaches. We implemented and evaluated a small-scale real-world mobile agent based campaign for pedestrian flock detection. The simulation and evaluation results show that mobile agent based campaigns produce comparable results with less energy consumption when the number of agents is relatively small and enables in-network data processing with sharing of data and task results with insignificant overhead.

Keywords Distributed computing · Multi-agent systems · Mobile computing · Mobile agents · Mobile crowdsensing

✉ Teemu Leppänen
teemu.leppanen@ee.oulu.fi

Yoshito Tobe
yoshito-tobe@rcl-aoyama.jp

Kaoru Sezaki
sezaki@iis.u-tokyo.ac.jp

1 Department of Computer Science and Engineering, University of Oulu, Oulu, Finland

2 Institute of Industrial Science, University of Tokyo, Tokyo, Japan

3 RealWorld Communication Laboratory, Aoyama Gakuin University, Tokyo, Japan

1 Introduction

With the introduction of smartphones into our everyday lives, new types of location-based and context-aware applications and services have emerged. Previously unobservable phenomena can be addressed by involving citizens in participatory sensing campaigns, where local knowledge of communities and special interest groups can be exploited at different scales for common good [4,5,26]. Crowdsensing applications create participatory sensor networks, where people take active roles with their personal mobile devices. Human intelligence and social interactions are leveraged to gather and analyze context-aware and semantically complex information and to share this knowledge of urban phenomena [5,19,31]. People and their situations are utilized to maximize the data quality, coordinate crowdsensing campaigns and delivering personalized services [5,13,26]. Sensor network deployments no longer require fixed infrastructure, but physical proximity is required. Human mobility patterns, everyday actions and willingness to participate opportunistically extends the data collection and sensing coverage with possibly large numbers of devices [1,4,5,18,26,36].

The unprecedented variety and scale of crowdsensing applications introduce challenges for systems design and development. Location-based and multimodal real-world data needs to be incorporated into campaigns and integrated with social networking services and existing systems to increase data utility [8,19,31,38]. Campaigns can't be designed without the consideration of human-oriented aspects, such as how participants interact with the application and the environment. In active (i.e. participatory) sensing, participants actively opt-in to campaigns and perform required actions, whereas in passive (i.e. opportunistic) sensing, the applications decide who meets the requirements and data is collected automatically [20,30]. Applications can be categorized into: (1) personal, involving data that the participant considers sensitive; (2) social, involving data that is shared within the social groups of the participant; and (3) public, involving data that is shared with everyone [4,27]. A crowdsensing software framework then coordinates the participants to cooperatively run the campaign and subsidize each other's efforts. Rules for participation and data collection need to be flexible, as different campaigns require different types of data. Incentives, monetary or otherwise beneficial to the participants, are to be introduced [19,20,31]. Participants' individual privacy needs are to be considered [14], where data ownership must remain with the participant [47], but on the other hand, collaboration through social interactions is to be facilitated [31]. In this respect, smartphones as the personal communication devices have become suitable platforms for crowdsensing. Smartphones are equipped with various physical sensors, include powerful microprocessors, have sizable data storage and communicate in short-range and over the Internet [18,19,31]. Participant devices are then programmed for manual or automatic and context-aware data collection.

Currently, no generally applicable solution for easy-to-follow mobile crowdsensing strategies with supporting system architecture exists, that can be easily verified or evaluated [7,20]. The existing approaches are either self-contained targeted applications, mobile task offloading frameworks or cloud-based data collection and analysis platforms. In this work, we present multi-agent systems (MAS) with mobile agents that provide decentralized and autonomous crowdsensing campaign execution and coordination. In general, software agents act on behalf of system entities, execute their tasks autonomously, react to the context in which they are operating, abstract heterogeneous resources and facilitate cooperation with agent and non-agent entities. MAS have been adopted for crowdsensing applications [48] and to provide application services for crowdsensing frameworks [23], where resident agents perform sensor data validation, information extraction, incentive generation and information sharing.

Service-oriented MAS architecture for mobile crowdsensing was presented in [24], where application-specific resident agents run crowdsensing services in the participating devices and mobile agents are utilized for augmenting the services of resident agents and transferring their accumulated results.

In this paper, we present how crowdsensing campaigns can be designed and implemented as a whole with mobile agents, that operate autonomously as a MAS in the opportunistic network of participants' devices in a given location within a given timeframe. Campaigns are designed through different roles and their interactions. Mobile agents realize these roles, negotiate the role execution with available resources in the devices, considering participants' individual privacy constraints and compensating them for their resource consumption and discomfort. The role-based tasks of mobile agents encapsulate data collection, filtering and pre-processing, feature extraction, event detection, data quality analysis, anonymization and sharing of data and results of the task in real-time. Real-time information sharing facilitates campaign execution and co-operation towards the campaign goal, while maintaining a selected global measure, e.g. energy efficiency. Energy efficiency is one self-evident measure that alone justifies the use of mobile agents as campaigns run on battery-operated smartphones. Moreover, through mobile agents, MAS facilitates concurrent execution of multiple campaigns without explicit crowdsensing application deployment that exhibits limited cooperation capabilities.

We discuss novel use cases for mobile agents in crowdsensing, their software requirements and campaign design issues. We present an implementation for mobile agent-based crowdsensing software framework over a resource-oriented architecture, based on our previous work in [33–35]. The framework does not provide a complete solution for MAS-based crowdsensing with mobile agents, but we develop the idea and lay the groundwork for advanced MAS features, such as sophisticated campaign control algorithms and negotiation capabilities of mobile agents. We simulate extensively different aspects of mobile agents in crowdsensing in comparison with the existing crowdsensing approaches. Lastly, we implemented a small-scale real-world crowdsensing campaign for pedestrian flock detection with mobile agents, executed in the software framework. The simulation and real-world evaluation results show, that mobile agent-based campaigns produce comparable results and target area coverage with reduced total campaign energy consumption with a relatively small number of mobile agents and increase data utility, i.e. sharing data and task results in real-time in the system, with insignificant overhead.

The rest of this paper is organized as follows. In Sect. 2, we present current challenges in crowdsensing based on the literature. Section 3 discusses how mobile agents can address these challenges and presents novel use cases for mobile agents in crowdsensing. Section 4 presents the proposed software framework, which is then evaluated by simulations in Sect. 5. In Sect. 6, we present and evaluate the small-scale real-world mobile agent-based crowdsensing application. In Sect. 7, we discuss the related work. Section 8 concludes the paper with discussion and considerations for future work.

2 Motivation for this work

Current crowdsensing[1] solutions and challenges have been considered in [1,4,5,13,14,18,20,26,27,29,31,36,38,48,52,54]. We discuss these challenges and propose mobile agent-based solutions, listed in Table 1, that are considered in detail in the next sections. Selected properties

[1] As the terms crowdsensing and participatory sensing are closely related, with the above-mentioned differences, we utilize the terms interchangeably.

Table 1 Current challenges in mobile crowdsensing and mobile agent-based solutions

Challenges in systems and campaigns	
Campaigns are coordinated centrally	Campaigns considered distributed multi-agent applications
Campaigns require different types of involvement and data collection styles	Multiple campaigns co-exist in MAS without explicitly considering one another
Unified system architecture for all campaigns that is integrated with external systems	Mobile agents facilitate data sharing and abstract data sources with common interfaces
Easy campaign deployment	Mobile agent-based roles and sensing tasks deployed into existing infrastructure
Campaigns need to be monitored in real-time	Mobile agents can detect, report and react to issues in campaign execution
Self-contained applications with no interaction capabilities	MAS's facilitate cooperation with in-network real-time data processing and result sharing
Data quality issues and incomplete data	Mobile agents perform data analysis and provide feedback to campaigner and participants in real-time
Human-related challenges	
Participant recruitment	Mobile agents monitor resource availability of participants and assess their previous performance
Participation rules need to be flexible	Mobile agents automatically detect match between campaign and participant contexts
Leveraging information from social interactions	Mobile agents integrate social networking systems as data sources for context-aware data
Participant commitment	Participant performance evaluated in real-time in location
Participant compensation	Mobile agents estimate fair compensation for resource consumption and discomfort in real-time
Data available in short intermittent periods	Mobile agents detect events in real-time in location to collect data
Data privacy management	Mobile agents encapsulate data, expose only selected and negotiated data features
Device-level challenges	
Heterogeneous user devices and data types	Mobile agents abstract device heterogeneity and different types of sensor data
Crowdsensing tasks should not interfere with smartphone normal use	Mobile agents react to smartphone use context
Battery-operated device with limited capabilities	Mobile agents can aim at reducing energy consumption
Task communication energy consumption	Mobile agents reduce amount of transmitted data with in-network data processing
Opportunistic data transmission	Mobile agents can react to changes in system and operate locally

of the proposed solutions are simulated and later considered in a real-world prototype in this paper.

Crowdsensing campaigns require, on the one hand, human knowledge, cognition, perception and social interactions, on the other hand, machine intelligence for data mining, computing and machine learning. Campaings are centrally coordinated, synchronized in location or time and facilitate systematic data collection embedded near the target areas or contexts, where data can be directly collected and analyzed on real-time. Crowdsensing campaign design considerations include the required degree of involvement and the style of data collection, e.g. is the campaign opportunistic or participatory. Widespread everyday use of smartphones offers better sensing coverage that exploits human mobility. Mobility patterns incorporate sociality and spatiotemporal "hotspots", where "social sensors" that observe human behavioral patterns are beneficial. Community models can be integrated into the campaigns to exploit this social intercourse. Leveraging this information into the campaigns is a challenge, but it becomes possible to have a broader perspective of the mobility patterns, activities and social interactions of individuals, and thus increase data quality with less variable results. Nevertheless, humans introduce randomness with their diverse actions, interests and goals. For this reason, collected data can be more related to people-to-people or people-to-environment interactions rather than to measuring physical phenomena systematically. Participants may even falsify the data while adapting their behaviors towards the campaign goal or for personal benefit. Campaigners can't control this behavior other than rewarding participants and utilizing participant profiles based on their performance monitoring. Participants' suitability for campaigns is to be assessed, where individual diversity increases in larger groups of participants.

Crowdsensing applications run in the background in the participant's personal devices and automatically collect data or opportunistically prompt for data entries, whenever application requirements match that participant's context. Often data collection contexts and data quality requirements can be met only for short and intermittent periods. Participant-related latencies in data collection and dissemination are inevitable due to their behavior. Different requirements for data source selection for the required context, coverage, resolution and data granularity are set for each campaign. Sensor calibration, data processing, verification, anonymization and context-aware decision-making introduce further computational overhead in the device. Other issues include incompleteness and quality of the data. Data can be validated and quality monitored with entry verification by the user and with real-time data processing in the device, such as context-aware filtering or semantically interpreting the data. Data are tagged with timestamp and location information indoors and outdoors, that increases its size and the amount of transmitted data. Data sharing with the other participants or social media applications can expose this refined data for common benefit, with increased communication energy consumption.

Campaigns rely on "people-powered" and heterogeneous devices carried by participants during their everyday actions. Devices, such as smartphones, are equipped with similar types of sensors, whose intended purpose is to elevate the user experience of the phone. The devices are programmed for either manual or automatic and context-aware data collection. Mobile and battery-operated devices bring limitations also with different sets of integrated hardware components and varying CPU power, in-device memory, and communication interfaces. Sensing is a low priority operation on the smartphone, thus sensing tasks should tolerate interruptions in the execution and should not interfere with the normal use of the phone, e.g. deplete the device battery or manage wireless connections while ignoring other applications.

Moreover, most crowdsensing applications are self-contained without interaction capabilities and developed for different operating systems and hardware platforms.

Ultimately, unified system architecture with standardized interfaces is needed to facilitate a broad range of campaigns, co-operation in data collection and analysis and to integrate external data sources into the campaigns, such as social media applications and Web services. In the crowdsensing systems, data transmission is opportunistic without infrastructure-based management and relies on the mobility, goals and social connections of the participants. Vast amounts of sensor data are generated and transmitted, thus large-scale storage capacity for historical data is needed for example in a cloud platform. Cloud platforms scale up, offer large-scale data fusion and analysis capabilities, integrate into external systems and can expose campaign results over the Internet.

3 Mobile agents for crowdsensing

In this section, we discuss how mobile agents can be utilized in crowdsensing and can provide solutions for the above-mentioned challenges. Mobile agents have been widely studied in distributed systems and in the context of wireless sensor networks (WSN). Generally, communication costs are reduced and network lifetime is increased when context-aware preprocessing and data redundancy removal are distributed and moved to the data source [10]. Robustness and fault tolerance are improved as tasks are executed autonomously and asynchronously taking into account local circumstances. Software components can be deployed dynamically in runtime, which facilitates software adaptation and system evolution, when mobile agents' tasks are updated instead of software in hosting devices [32].

3.1 Crowdsensing campaigns

A general model of coordinated crowdsensing campaign is described in [44] with the following stages: (1) campaign creation; (2) participant recruitment; (3) campaign execution; (4) results verification; and (5) attestation, privacy-aware sharing and publication of results. A campaign is created based on a specification, that defines required sensing context, regions of interest in space and time, campaign budget, participant selection criteria and campaign starting and ending criteria. The specification is used in participant recruitment to provide maximum utility and expected level of participant commitment, according to the campaign budget. Campaign execution component then orchestrates data collection by tasking the participant devices and keeping the participants in the loop of task progress. However, participants regulate their own participation. Watchdog component estimates the individual effort of the participants and rewards them accordingly. A verification mechanism checks the data quality with location attestation or with cross-checking, where participants may prefer anonymity. The campaign results are published with selective sharing that maintains negotiated privacy levels.

MAS are designed in terms of agent roles and their interactions. Roles provide abstractions that enable high-level interaction capabilities between cooperating agents, allow agents to specialize in behavior, and reduce the number of agents competing for any task [6]. In MAS, roles interact with other roles and external data sources, which establishes system configuration and abstract model of data flow. Complex tasks can be then defined through cooperating roles. For each role, its data inputs, outputs and associated computations are described that define its view of the system. Then, MAS is coordinated through the role outputs synchronously or can be left operating asynchronously without further consideration

by the campaigner. Agent interaction protocols need to be defined to enable interactions and resource access with a uniform interface.

We consider crowdsensing campaigns designed and implemented as a MAS, where the goal is generally data collection about specific real-world phenomena. Campaigns are designed and organized as roles and their interactions that correspond to the campaign requirements and goals. In crowdsensing, the minimal role required is sensor data collector(s) for the required types of data, which in the existing solutions is implemented without the agent abstraction. In an unified system architecture, roles cooperate and interact with other roles even outside their campaigns. Roles could be designed as general data sources for different campaigns to some extent, which facilitates cooperation further. This is difficult to implement in self-contained applications or crowdsensing frameworks without interaction capabilities. An additional campaign-specific selected global measure can be introduced, e.g. maximum sensing coverage or energy-efficiency with limited coverage, that is taken into account by the roles in executing their tasks. The participants' devices are no longer operated by a single authority, but opportunistically utilized while considering the participant's individual context. Now, the responsibilities of the role must have limits, as its operations may be suboptimal for some participants. Therefore, participants must have means to negotiate and control their own participation. The underlying crowdsensing system is opportunistic and in continuous transition due to human mobility and resource availability that can't be modeled explicitly, thus campaign organization model is abstract.

Group abstraction is beneficial in the campaign design and coordination. Roles are organized into groups that are defined spatially, temporally, contextually or based on interaction patterns of roles. A role can belong to several groups simultaneously, which are addressed separately. Efficient campaign execution requires management of the group structure, roles and their memberships, where memberships can be based on a number of well-known clustering algorithms. Groups select a representative agent, that solely communicates outside the group. This role is particularly useful for campaigns with large geographical target area, large number of participants and large amount of data. The representative can adopt the role of data aggregator that buffers data, removes redundancies and extracts information from the data received from group members, before disseminating it further. Communication load is reduced and security increased if group members only communicate with each other and if data processing is distributed and partitioned between members. The representative role circulates among the group members to distribute the extra energy consumption. The challenge lies in implementing an efficient clustering algorithm for opportunistic networks with variable group count and unequal group sizes for heterogeneous mobile devices. As an example, representatives are selected based on the lowest movement speed in relation to the other group members or history of the members' contact patterns that assumes Wi-Fi-based wireless communications [22].

Mobile agents embrace different campaign-specific roles and encapsulate the functionality and interactions of the role as a single computational unit. Now, campaigns are realized by collaborative actions of mobile agents in the MAS. Campaigners initiate and run crowdsensing campaigns by injecting mobile agents into the system. Mobile agents facilitate the online model where sensing tasks arrive at the system in runtime, as opposed to an offline model where all the tasks are known at the time of scheduling. With multiple simultaneous campaigns running in the same system, mobile agents can identify common goals for their tasks, reuse the data and features exposed by other agents, which distributes and could reduce the resource utilization in the campaigns. The negotiation and conflict resolution capabilities of mobile agents in a MAS assists in co-operating towards campaign goals while maintaining the global measure. Completely autonomous and distributed campaign control requires

reflection capability and reasoning about the campaign execution. During the execution, mobile agents autonomously react to the local situation, negotiate with the devices to utilize their resources and access global system resources and migrate based on their task and the device context. Several modes for campaign operation can be identified. First, the campaigner performs one-time resource lookup to the system for the participant selection criteria, creates and injects the mobile agents into the system with the configuration available at that time. This also facilitates small-scale tasks, i.e. applications that require occasional data from a target location [1]. Secondly, the campaigner monitors the system for available resources in its sensing context, and injects mobile agents into the system whenever resources become available. This maintains the critical mass of devices for the campaign. Thirdly, mobile agents feature various levels of autonomy in their operation, can clone themselves or spawn new agents, for example to maintain the critical mass. It becomes possible even for the mobile agents to update their roles in the sensing context or assume different roles in campaigns.

Mobile agents have been used to implement different data routing algorithms in WSNs. In crowdsensing, the challenge is that device availability can't be guaranteed, thus data routing decisions need to be dynamic. Moreover, routing schemes change between applications that differ in the characteristics of human mobility, uncertainty of future connectivity, availability and heterogeneity of the participating devices, thus no single routing scheme is optimal [13]. A method for energy-efficient data routing, based on participant flocks, was presented in [22]. Flocks are detected based on the recent histories of wireless connectivity between the participants, who then form a cluster among their connections. Data are first shared within all devices in the cluster. Whenever neighboring flock is detected, data are exchanged between the flocks. The neighboring flocks are detected by a subset of devices in the cluster to reduce the energy consumption of contact probing. Moreover, in [50], several data routing algorithms in delay-tolerant networks were evaluated for opportunistic sensing with mobile and static sinks.

3.2 Sensing tasks

The sensing tasks for crowdsensing campaigns are defined in terms of sensing context and required in-network data processing. Tasks are operated by mobile agents as a part of their role, where data sources are incorporated into the agent composition. A challenge is that crowdsensing data is typically noisy [11]. Data processing includes clean-up, feature extraction, event detection, quality assessment and privacy protection. Moreover, real-time in-network data processing facilitates exposing refined data and task results to the other system components. For example, energy consumption can be reduced by sharing data from energy hungry sensors, such as GPS receiver, with local short-range communications between participant devices [17]. Multiple data processing tasks can be included in a mobile agent that each expose different representation of the same data. These tasks may utilize different resolution in data collection, such as intelligent oversampling and subsampling, in specific locations. As data are collected by different sensors in local environment, the amount of transmitted data can be further reduced within a group, when the group representative (i.e. data aggregator) adaptively samples the data and shares its result. This way it is possible to avoid data round-trip to the backend system.

Mobile agents incorporate different sensing modes, such as manual, automatic or context-aware. In manual mode, the mobile agent lives in the hosting device but only operates when new data is available, e.g. in case the participant initiates sensing or inputs data. In automatic mode, the mobile agent controls the sensing task execution and sensing parameters, for example starts and stops the sensing or changes the sampling rate. In context-aware

mode, the mobile agent reacts and operates according to the specified context, for example by starting sensing based on detected events. Mobile agens facilitate both participatory and opportunistic sensing approaches. Users opt-in to participatory tasks by allowing an agent to migrate into their smartphones and make data available as negotiated with the campaigner. Opportunistic approaches exhibit automated sampling and application-specific data processing, depending on how the participants agree to the sensing context. To distribute the task execution, the agent's migration policy can feature fixed or non-fixed itineraries. However, resource availability and sensing coverage can evolve in runtime, which dictates how mobile agents operate and migrate in MAS by taking into account sensing contexts and participant set constraints. The dynamic availability of components in the devices, such as specific sensors, can't be guaranteed at all times as they may be in use by other applications. To avoid this bottleneck, migration should be based not only on the selection of the best matching device for a context but also include its groups' members. Also, mobile agents can reroute other agents [16].

Task-based interactions of the mobile agents include controlling local and external resources, such as actuating a sensor, and sharing data and task results. Sharing is achieved through their states, which are returned for a request or uploaded to the state to the campaigner by the agent. The state contains the current result of the task, but can also include real-time information about task execution, device resource availability and information about the participant's performance and context. This enables autonomous cooperation with other agents and non-agent system components. Other agents can react to state changes, detect their task-specific events and adjust their behavior accordingly. The task result can be utilized by the campaigner to control the campaign. Moreover, some tasks are executed in response to an event, when a participant exposes particular context or appears in target location. Data sharing is based on specific queries, either one-shot or continuous [45]. One-shot queries include single point-of-interest queries and spatial aggregation over a region or a trajectory. Continuous queries include monitoring and event detection tasks. For one-shot queries, the devices can receive compensation for each data sample and for the loss of privacy. For continuous queries, a budget is set and the question is which devices to use, where and when.

A request-based method for data sharing is presented in [39]. Collected sensor data is stored in the device to save communication energy, but devices send real-time notifications of their current location to the framework to facilitate location-based data queries. Data is uploaded to the framework only on request. In [45], a data acquisition framework enables sharing of the sensors and data between multiple simultaneous queries. The campaigners submit queries to an aggregator component, that selects appropriate participating devices based on the value of the data for multiple queries, common data requirements and the cost of obtaining such samples. The queries include requested data sampling rate and a valuation function to assess the quality of the collected data samples.

3.3 Participant-related aspects

Successful campaign execution requires consideration of human-related aspects that include identifying the right set of devices and participants [8,19,28,31,43]. Participants are recruited based on their availability, activity, location, context, events and history [8,23,25,28,43,50]. Reputation can identify and promote certain participants for on-going campaigns. Participants prefer long-term automatic tasks over short-term manual tasks and task completion ratio is high once the task is accepted [8]. For opportunistic sensing campaigns, it is beneficial to consider also participants whose context matches the campaign requirements only partially, provided that the task coverage then increases [50]. Participants may have different

understanding about the task goals, in which case, the real-time data verification and quality assurance provides feedback for the participants in campaign runtime.

Mobile agents embrace the task of monitoring campaign execution through the behavior of the participants and observed data quality. Unwanted behavior of participants can be detected by mobile agents, which are able to suggest real-time modifications to the in-situ data collection. Participants' performance and reputation are evaluated in runtime, based on different policies, such as the attendance of the participant or recency of the participants' actions [8]. The reputation is presented, for example, in a ranked list [8] for each task and maintained automatically through the exposed mobile agent's states. If a participant cancels sensing tasks in runtime or the individual constrains for participation are met, there are several options to continue task execution. The agent state is returned to the campaigner as its final result. The agent can migrate from the device to continue the task execution within the group members or other available devices. New agents can be injected or spawned in runtime. Due to the autonomy of mobile agents, it is not required to have a priori knowledge of the context or future goals of the participants for successful completion of a campaign.

Sensing coverage is improved around places that people frequently visit, but coverage can be addressed by fostering increased compensation for more frequent visits to the target location. Participants can be persuaded to change travelling routes for places that are relatively close to their location [11]. However, unpredictable mobility patterns introduce randomness [11], although human mobility traces have been found to be stable enough for meaningful predictions of future traces [55]. If campaigners have no access to the location information about participants, exposed or publicly available trajectory histories can be utilized to build a mobility model and estimate future locations [40]. To increase accuracy of the model, spatial and temporal information about the participants can be reported to the system in real-time to control the campaign execution and update the mobility model. To improve the target area coverage, reverse auctioning based on the participants' location and available campaign budget can be utilized as in [25]. Mobile agents can guide the participant movement, while protecting participants' privacy by adding uncertainty to the location information.

If participants don't directly benefit from the data or the application itself, compensation for their resource consumption needs to be considered. Compensation can be measured in terms of energy consumption of crowdsensing application execution, data transmission and required actions and interactions, but also based on some user discomfort measure [28,49]. Compensation costs are considered private information for each participant, which can cause issues for the campaigner if participants misreport their actual costs [8]. Previous work demonstrates reverse auctioning to solve this issue [28]. The campaigner uses a priori metrics to calculate the compensation, such as task progress, received data quality or role of the participant. Additionally, campaigners can utilize historical data of target areas to estimate the demand and supply of data on that location, i.e. frequently visited location delivers less compensation in return than an infrequently visited location [49]. Mobile agents can measure actual costs fairly and campaigner issues the individual compensations according to resource availability and negotiated contracts. At the same time, agents react to changes in user and sensing context, which may lead to decreased compensation.

3.4 Privacy management

The data collected in people-centric applications can reveal the personal habits or behavior of the participants. Location traces, time traces and data upload schedules reveal additional information. This data might be utilized without consent, once exposed to the system [1,12].

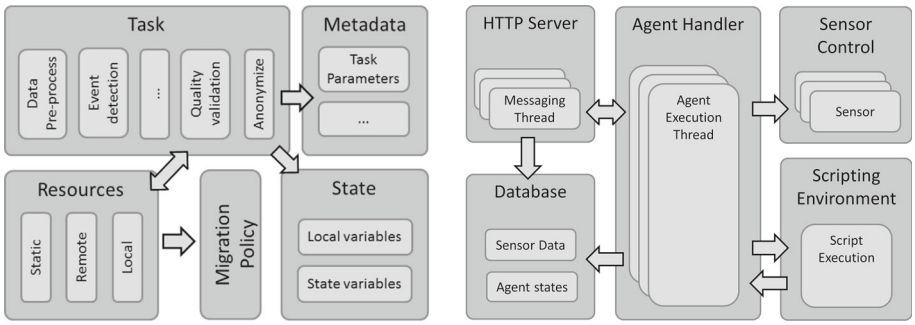
Sensing tasks have their own parameters, where deployment of the task to the system can reveal the identities of the participants if a rare sensor type or a particular place of interest is requested [12]. Users may not be aware of which applications are running in their smartphones and what kind of data and location information they are collecting. Moreover, long-term data storages in the backend systems also face diverse types of attacks [27].

When potential participants are made aware of these possible ways of intrusion into their privacy, they may become reluctant to contribute to campaigns [12,14,45]. Therefore, participants should be able to express personal privacy preferences and negotiate the appropriate privacy level for each campaign that are then included in the campaign execution from the data collection to the analysis and long-term data retention [47]. After all, in the end, only a subset of data may be needed by the end users of the campaign results [12,47]. Participants should explicitly negotiate the type, temporal resolution and granularity of information that is collected. They should be able to exclude sensitive locations, such as home, from the campaign. They should control access to their data, and moreover, what they are willing to share with whom and for how long [5,12,47]. Here socio-cultural and contextual differences also impact the individual perception of data sensitivity [12]. Nonetheless, in opportunistic sensing this could be concern as users may falsify the data or perform biased sampling for personal gain [1,12]. Crowdsensing application developers should implement negotiation means for data sharing, where sharing practices may vary across campaigns. The crowdsensing platform should facilitate visualizing the data with campaign-specific interpretations to help the users understand what is actually revealed about them [1,47]. As an example, AnonySense [14] is a framework for crowdsensing that protects the privacy of participants and guarantees integrity of the received data, based on a communication channel mixing between the participants and the data recipient that during upload de-links data from its sources.

Mobile agents negotiate between the participant's privacy requirements and the campaign data requirements, while monitoring that privacy rules are followed and possibly automatically adjusting sensing parameters. Mobile agents encapsulate data for short-term storage and expose only a subset of data or its features. Data processing algorithms can include data perturbation, tailored to the specific task and campaign. Mobile agents can implement spatial cloaking and pseudonymity with migration, obfuscating the data source and the data processing host. Context resolution is to be considered also, as the users may not want to reveal too much information about their identity or location [4]. On the other hand, maintaining participants' reputations requires analyzing application-specific detailed information. If the participant is simultaneously participating in several campaigns or hosting multiple agents, each agent must operate without any knowledge of the existence of others. Data sharing interfaces facilitate privacy protection requirements, despite of the data location, such as local data in the phone or remotely retrieved data. Resource access keys and authentication methods can be incorporated into the mobile agent. The exposed information can be further visualized in the application user interface through the agents' states, so that the user can verify the information about to be shared. However, in emergency situations [12], for example, if critical events are detected from the data, these privacy rules can be overridden by the mobile agents.

4 Implementation of mobile agent-based crowdsensing

In this section, we describe the internals of mobile agents and a software framework that enables utilizing mobile agents in crowdsensing. This work is based on the work in [3] and our previous work in [33–35]. The presented architecture and the framework are not fully FIPA compliant.



(a) Mobile agent **(b)** Mobile agent execution environment

Fig. 1 Internal architecture and components

We adopt a resource-oriented architecture, which seamlessly integrates the framework and campaigns into the Internet through RESTful interfaces. Key abstraction is a resource, that is something having value in the campaigns. We identify the following resources: (1) participants' devices as sensing task and agent execution platforms; (2) integrated physical sensors, including GPS receiver, Wi-Fi radio, camera and microphone; (3) raw and refined sensor data; and (4) mobile agents through their states. Each resource is uniquely identified and addressable with an URL. HTTP is utilized as a standardized universal communication protocol, thus each system component needs to run an HTTP server. HTTP methods and their corresponding response codes are utilized as generic primitives in communication within the system, including intra- and inter-agent communications. This seamlessly integrates external Web services as data sources for the campaigns. Moreover, HTTP includes content type negotiation and security mechanisms.

4.1 Internal architecture of mobile agents

A mobile agent has the following properties: state, implementation, interactions and a unique identifier [10,16]. The state abstracts the agent functionality and represents the agent in the system. Agent implementation defines its its behavior and interactions enable the agent to reason about its environment and to co-operate. A unique identifier is needed to address the agent globally. The agent can also have metadata related to its operation and interactions, such as security and privacy attributes.

The internal architecture of the mobile agent in this work is shown in Fig. 1a, where arrows depict dependencies between the components, e.g. the state is updated by the task. The tasks are separated into different functions, which implement task-specific data processing algorithms and their interactions. Tasks can control sensing parameters, such as sample rate, and actuate the physical sensor. Task functionalities are labeled in the agent composition and can be implement in any programming language, if provided with language-specific identifiers. This facilitates separation of concerns and enables modifying the agent functionality in runtime and to create higher-level plans for MAS. Agent composition can include a reference, instead of code block, for on-demand retrieval of the task code [34].

The resources needed by the task are listed in the resource segment. Resources include data sources and system components, which the task controls and interacts with. For each resource, the composition includes a reference (i.e. URL) that can have additional query parameters for resource specific operation, such as a time period for the retrieved data. We

have defined three types of resources in [34]: (1) local resources that are hosted by the sensing device; (2) remote resources that are external to the device; and (3) static resources that are invariable remote resources for the lifetime of the agent. Local resources determine to where the agent migrates, whereas remote resources are accessed in each iteration of the agent execution and static resources are accessed only once during the agent execution. As an example, the barometer sensor in the agent hosting smartphone is a local resource, current temperature from an infrastructure service is a remote resource and average temperature of yesterday is a static resource.

This separation to different resources types is justified for resources access. Mobile agents reason about resource access and their migration, by analyzing each interaction before it is conducted. The decision to migrate and the next host are selected based on some local measure, such as the amount of collected unprocessed data on the device, as shown in later in this paper. This measure tries to minimize communication energy consumption and to avoid data loss. When the agent composition is modifiable, the resource type can change. However, a resource may be hosted by a device that can not execute mobile agents, thus the resource must be considered as remote.

State segment exposes the current results of the agent task, that are updated after each time the task is executed. The campaign designer can select what additional information is included in the state, if any. Multiple representations of the data are facilitated, as well as exposing the task execution- and participant-related information. Task local variables that are retained across the multiple iterations of task execution, can be stored in the state, but are not exposed as a part of it.

4.2 Mobile agent execution environment

The software components needed in participating devices include: a tasking component to control and monitor the task execution, a data analysis component, a storage component for the data task results, a presentation component to visualize the results and system components for core network services.

With mobile agents, agent execution environment (EE) application is needed to run their tasks and handle their interactions. Internal architecture of the EE in this work is shown in Fig. 1b, where arrows depict the direction that data or control flows. EE contains an HTTP server for communications over the Internet, a component to handle mobile agents, a component to execute the agent task code (e.g. scripts), a component to control the sensors in the device and a database component to store sensor data and task results. EE handles data requests and agent-based interactions on behalf of the agents. Whenever external requests for data arrive or agent initiates a conversation, the EE creates new thread for it. This way, it can handle multiple simultaneous interactions. Sensor data in the database is augmented with timestamp and location information and is annotated for the particular task.

The EE executes agents' tasks in their own threads. Scheduling and execution of threads is done by application or operating system. First, EE parses the agent message and generates a runnable script for the task. Then, it runs the script, and during task execution, retrieves the required local and external data through the references in the composition. Also, during the execution, it disseminates local or external agent-based messages, dispatches event notifications and controls external components through the references. After task execution, the current task result is updated into the compositon. If required, the agent can also upload the result to the campaigner, which can monitor the campaign progress and control the task execution with this real-time information. Moreover, when results are stored by the campaigner, the participating devices need not to handle external requests out side the campaign.

When the agent makes decision to migrate, its task code, resource references and state are composed into a message that is send to the next host.

Tasks can start their own asynchronous conversations with other system components, based on their state, events and received messages, with specific agent communication languages. Same communication interface should be utilized for both local and external communications. We have defined HTTP methods and their response codes as the general agent interaction protocols, shown in Table 2. The method OPTIONS is used to get resource metadata, that facilitates reasoning about the resource access and agent migration. The method GET is used to retrieve resource representation, e.g. sensor data or agent state. The method POST is used to: (1) actuate a resource, e.g. turn a sensor on or off; (2) migrate a mobile agent between devices; and (3) register resources to the system whenever new participants or data sources appear. The DELETE method is used to remove a resource from the system that is useful when resources are utilized by some other application or the participant wants to control the usage of resources. In all requests,"Unauthorised" response can be returned in case of authentication failure.

This approach provides limited autonomy for the mobile agents. Agents are reactive, based on retrieved data and their internal state, and exhibit limited reasoning capabilities about their own migration and resource utilization. Agents can start new simple conversations with other system components. With this EE implementation, it is not possible for the agents to autonomously negotiate or control the campaign execution or create new tasks or goals.

4.3 Software framework

Software frameworks for crowdsensing require specific software components for infrastructure and for participant devices. The framework must support public and globally reachable resource naming, resource discovery, data management and large-scale in-network data analysis. The framework should utilize standardized communication protocols [1,4]. Mobile agent software frameworks require components to facilitate different agent behaviours and interactions within the system [16]: agent creation and dispatch, agent locating and tracking mechanisms, task synchronization, agent interaction protocols, agent EEs with general communication interfaces. Crowdsensing software framework adopts an opportunistic networking approach, where communication mode is peer-to-peer and the campaigner and participants are assumed to be at single-hop distance. This simplifies executing agent interactions and migration policies.

Figure 2 illustrates the software framework for mobile agent-based crowdsensing. A campaigner Web service controls and monitors the campaign execution. A distributed resource directory (DRD) [35] provides a directory service for the resources that are available for campaigns. For Internet and Web connectivity, each participating device should have an IP address. Each resource registers to the DRD in join-time with a resource description to participate into the campaigns. The resource description contains unique resource identifier and current address as URL, its type and semantic description and additional metadata, such as security parameters. Each EE performs periodic updates for its resources when there are changes, such as agent migration. This way, system components can perform runtime lookups for available resources. In MAS, one role could be a DRD peer that provides local copies of the directory.

To execute a campaign in this framework, a set of interaction protocols (Table 2) are required. "Request resource metadata" is used to reason about resource access and agent migration, "Query resource representation" to is used retrieve resource representation to the current host and "Request action" to send data or dispatch events. "Request resource deletion"

Table 2 HTTP-based mobile agent interaction protocols

Protocol	Method	Response	
Request resource metadata	OPTIONS	OK	Metadata
Query resource representation	GET	OK	Resource representation
Request action	POST	OK	Action performed
		Forbidden	Action has been refused
		Accepted	The action will be performed later
Modify agent behavior	POST	Created	New task has been created for the agent
Request resource deletion	DELETE	No content	Resource deleted from system
		Accepted	Deletion will be performed later, when resource becomes unused

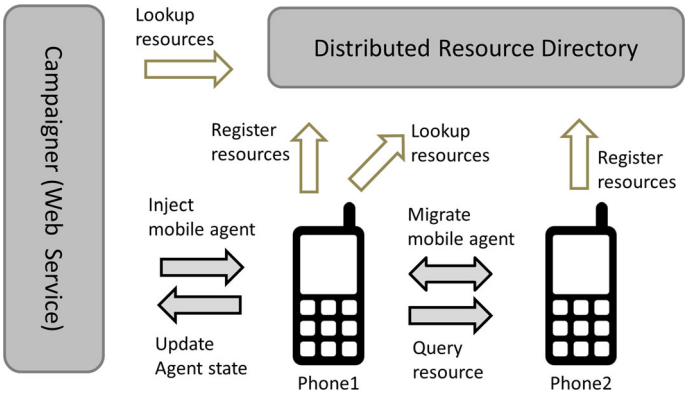


Fig. 2 Software framework to enable mobile agents in mobile crowdsensing

is used to remove a resource from the DRD. During campaign execution in this framework, campaign control and managing the mobile agents is achieved through the uploaded task results and execution parameters from participants to the campaigner in runtime. With mobile agents, campaign control can be minimal, where the group abstraction assists in both resource lookup and control. The campaigner is responsible for maintaining sufficient number of mobile agents, taking into account the currently available resources. The campaigner should also monitor the results to determine when the campaign is finished. This contributes to the robust, flexible and scalable execution of the campaign. However, generally MAS facilitates sophisticated campaign control algorithms that can are operated by the agents, e.g. with the "Modify agent behavior" protocol.

In highly distributed multi-agent application, resource discovery is challenging due to mobility and, in crowdsensing, unpredictable behavior of the participants and resource availability. As mobile agents migrate, existing bindings to the task's resources are constantly broken. Automatic rebinding at the update of reference would assist with this problem, but due to the high mobility, it could create significant communication overhead. Therefore, for crowdsensing, each time a resource access is needed, a lookup and rebinding are performed. If a resource has disconnected or disappeared, the lookup fails and access is not performed in this iteration of the mobile agent execution. Eventually, if resources can't be accessed, agents should return their results to the campaigner and delete themselves, or migrate away from the device.

5 Evaluation

We designed a set of campaigns to evaluate mobile agent based crowdsensing in comparison with "traditional" approaches as reference campaigns. The campaigns demonstrate both spatial and temporal aspects of crowdsensing. In [23], crowdsensing MAS performance and overhead was evaluated in terms of execution latencies, battery consumption and communication energy consumption. However, the results are not compared with other crowdsensing approaches. Previous work has demonstrated energy efficiency of mobile agent-based WSN that reduces the amount of transmitted data [10,16].

In this evaluation, we consider total energy consumption in the campaigns with and without mobile agents. We measure the time between participants' visits to the same point-of-interest,

defined as inter-cover time in [36]. Evaluation with mobile agents consider computation overhead of agent execution and inter-agent communications. Real-time data utility is considered, that refers to sharing collected data and agents' task results in runtime in the campaign. For communications, we consider total amount of transmitted data, number of messages and different message types. We also measure the number of participants in campaigns and how that influences campaign execution.

We assume that a crowdsensing campaign has already been created by the campaigner and the starting criteria have been met. The campaign defines a target area, where the participants move, carrying their smartphones with them, until they exit the area or campaign end criteria is met. The participant devices are aware of the sensing context and opportunistic store-carry-and-forward method in used data collection: (1) the devices store the collected raw data until a mobile agent migrates into the device; (2) data is processed by the mobile agent; and (3) agent task result is uploaded to the campaigner after execution.

We utilize two migration policies on the campaigns. Firstly, mobile agents migrate into the devices from the campaigner when the devices appear in the target area and live in the device until the end of participation. Secondly, the campaigner injects a number of mobile agents that roam freely, migrating in each round into the devices with most collected raw data. This guarantees that every migration is justified in terms of communication energy consumption, i.e. retrieving the data to the current host is more expensive based on the resource sizes. This policy prevents data loss in case the device fails or leaves the task, because it tries to maximize the amount of processed data. For resource access and migration, agents are aware of the amount of data in their possible migration hosts, as they receive this information when uploading task results to the campaigner.

The campaigns listed in Table 3 are considered in the evaluation. Campaigns T1_1, T1_2 and T2 are the reference campaigns. Campaign T1_1 demonstrates self-contained application, where T1_2 augments the application with adaptive sampling. As adaptive sampling example, the participants receive information from the campaigner about redundant locations in the participants future path. In these locations, sensors are turned off and task results are not calculated nor uploaded to the campaigner. Campaign T2 demonstrates a self-contained application, which uploads all data to the campaigner at once at the end of its participation. Campaigns T3_1-T7 are based on mobile agents and designed with increasing complexity. Campaign T3_1 demonstrates the store-carry-and-forward method, where mobile agents migrate into each participant's device when they appear at the target location as live in the device until end of participation. Campaign T3_2 adds adaptive sampling to this method. Campaign T4_1 demonstrates the effect of increasing numbers of roaming mobile agents, where T4_2 includes adaptive sampling in the agents' tasks. Campaign T5 demonstrates the effects of increasing the number of participants, without increasing the number of roaming mobile agents. Campaign T6 demonstrates simple MAS based campaign, where agents' states are shared and total overhead energy of sending and receiving is calculated. Lastly, campaign T7 demonstrates the effects of sharing raw data with varying campaign parameters such as number of mobile agents, number of requests per round and requested data items, in relation to each other.

5.1 Simulations

We conducted a set of simulations to evaluate the designed campaigns. Simulations were implemented with NetLogo[2]. In the simulations, we collected a number of attributes from

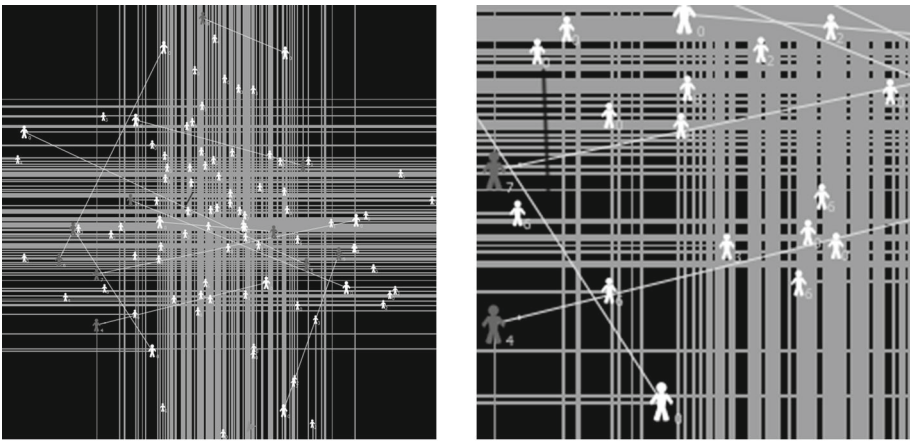
[2] https://ccl.northwestern.edu/netlogo/

Table 3 Description of the campaigns in the evaluation

Reference campaigns	
T1_1	(1) No mobile agents
	(2) Sensor data is collected at each cell and uploaded to the campaigner in real-time
	(3) One new participant appears at 50 % probability in each round
T1_2	(4) Application utilizes adaptive sampling
T2	(1) No mobile agents
	(2) Sensor data is collected at each cell, but uploaded to the campaigner exactly once when leaving the campaign
	(3) One new participant appears at 50 % probability in each round
Mobile agents-based campaigns	
T3_1	(1) Mobile agents migrate to devices when they appear at the target area
	(2) Sensor data is collected at each cell, processed by the mobile agent and uploaded to the campaigner in real-time
	(3) One new participant appears at 50 % probability in each round
T3_2	(4) Agents utilize adaptive sampling
T4_1	(1) Different number of mobile agents (1–90) roam and migrate to the devices with most new collected data
	(2) Sensor data is collected at each cell, but processed by the mobile agent when it migrates into the device and then uploaded to the campaigner
	(3) One new participant appears at 50 % probability in each round
T4_2	(4) Agents utilize adaptive sampling
T5	(1) Number of mobile agents is fixed at 50
	(2) Sensor data is collected at each cell, but processed by the mobile agent when it migrates into the device and then uploaded to the campaigner
	(3) New participants appear into the target area at increased rate
T6	(1) 1–20 mobile agents roam. 80 % of agents share their state with 20 % of agents at each round
T7	(1) 1–20 mobile agents roam and share 1–20 raw data items with 1–20 requests per round

the campaigns: total campaign energy consumption, inter-agent communication energy consumption, campaign completion time, inter-cover time, amount of lost data, mobile agent execution energy consumption, number of messages and message types and number of participants. A snapshot of on-going simulation is shown in Fig 3a, b, where the colour of unvisited cells is black and visited cells light grey. Participants are depicted with person icons, in which the adjacent label indicates the amount of raw data hosted by each participant in that round. The person icons change size when involved in agent migration and colour to dark grey when hosting an agent. Links depict agent migration.

The simulation parameters are listed at Table 4, which include simulation design parameters, campaigner set parameters, parameters related to the participants and parameters of the application and campaign execution. Wi-Fi is utilized as the communication technology, but we omitted the Wi-Fi scan and connection setup energy consumptions from the simulation. HTTP is used as communication protocol. Participants have already been selected and assumed to have full battery in their smartphones, as they appear in the target area at campaign-specific rate in random locations (normal distribution with empirical rule) at the edges of the area. Therefore, the target area centre draws more participants, corresponding



(a) The whole simulation area **(b)** Excerpt from the simulation area

Fig. 3 Visualizations of an on-going simulated campaign

Table 4 The parameters utilized in the simulated campaigns

Global parameters	
Target area	Grid with 400 × 400 cells, cell dimensions 0.5m × 0.5m
Time	One round is one second
Participants appear rate	Specific to campaign
Communication technology	Wi-Fi, unlimited bandwidth and uniform coverage, single-hop distance
Campaigner set	
Campaign launch criteria	20 participants available
Campaign end criteria	80 % sensing coverage of the target area
Participants	
Moving direction	Random: up, down, left or right in straight line
Moving speed	Random: 1–5 cells per round
Available phone energy	Unlimited
Application parameters	
Power consumption	Communication: 164 mW, radio awake at all times [51]
	Application: 5mW, a real-world prototype in Sect. 6
	Sensor: included in application power consumption
	Agent EE: 5mW, assumed
Data transmission energy	Participants divided into three groups: 1/3 of participants consume 5 mJ/kB, 1/3 consumes 20 mJ/kB and 1/3 consumes 40 mJ/kB in both sending and receiving data
HTTP protocol overhead	GET 370 bytes, POST 225 bytes
Composition size	Total size 535 bytes, state is 135 bytes
Sensor data item size	100 bytes with timestamp, GPS location and data value

to the "hotspots" and "sociality" of participants that are significant human behavioral factors [36]. Participants travel with random walking speed (0.5–2.5m/s), i.e, one to five cells in each round. The movement direction is fixed (up, down, left, right) following a straight line from an edge of the grid area towards the centre and finish at the opposite edge of the grid. In each cell, the participant travels, one sensor data item is collected.

5.2 Simulation results

We present the results of the simulated campaigns in Figs. 4a, 5, 6, and 7 and analyze each campaign below. Additionally, Fig. 4f shows the effect of different mobile agent execution energy consumption on the EE that is a significant factor in the real-world prototype, as discussed in Sect. 6. In the simulations, we utilized the measured real-world agent execution energy consumption. In the figures, the x-axis corresponds to number of mobile agents in the campaign except in Fig. 4b, where the x-axis corresponds to varying number of data requests, data items and hosts.

Campaign T1_1 All data is uploaded to the campaigner in real-time from a self-contained application. The energy consumption is quite high in comparison to other campaigns, as shown in Fig. 4a.

Campaign T1_2 With application-specific adaptive sampling, the communication energy consumption of real-time data upload is reduced significantly (Fig. 5). This has no effect on campaign completion time or inter-cover time, which relies on participant movement. No data is lost (Fig. 4e).

Campaign T2 This campaign consumes more total energy than the real-time data upload with adaptive sampling (T1_2), even though it only uploads data only once. But, this method is more energy efficient than uploading all data in real-time (campaign T1_1). No effect is seen in campaign completion time or in inter-cover time. No data is lost.

Campaign T3_1 When mobile agents become resident agents in the participant devices, we observe increase in energy consumption in comparison to campaign T1_1. This is due to the mobile agent execution overhead (Fig. 4f), even though the communication energy consumption decreases after local data processing (Fig. 5). No effect is seen in campaign completion time or in inter-cover time. No data is lost.

Campaign T3_2 With adaptive sampling, the energy consumption of resident mobile agents decreases significantly. Figure 5 shows decreases in both communication energy consumption and agent execution energy overhead. But still, this campaign consumes slightly more energy than campaign T1_2 due to the agent execution energy overhead. In comparison to campaign T2, about 25 % reduction is seen in total energy consumption (Fig. 4a). Yet, no effect on campaign completion time or inter-cover time. No data is lost.

Campaign T4_1 The energy consumption is very low with less than ten agents, even lower than the energy consumption of all reference campaigns. But it increases significantly when the number of agents is increased, almost to the level of resident agents in campaign T3_1. The reasons for this are increasing communication and agent execution energy consumption (Fig. 5). With a large number of agents, upload messages dominate the number of messages (Fig. 6) and the number of agent migration messages is reduced, as mobile agents become resident agents. The task completion and inter-cover time vary little, but are within 10 % range of previous campaigns. The number of lost data items is large with a small number of agents, due to the fact that the agents do not have enough time to migrate and handle all the collected data before the participants leave the campaign.

Campaign T4_2 Roaming mobile agents with adaptive sampling consume about the same amount of energy than the best reference campaign (T1_2). Energy consumption is

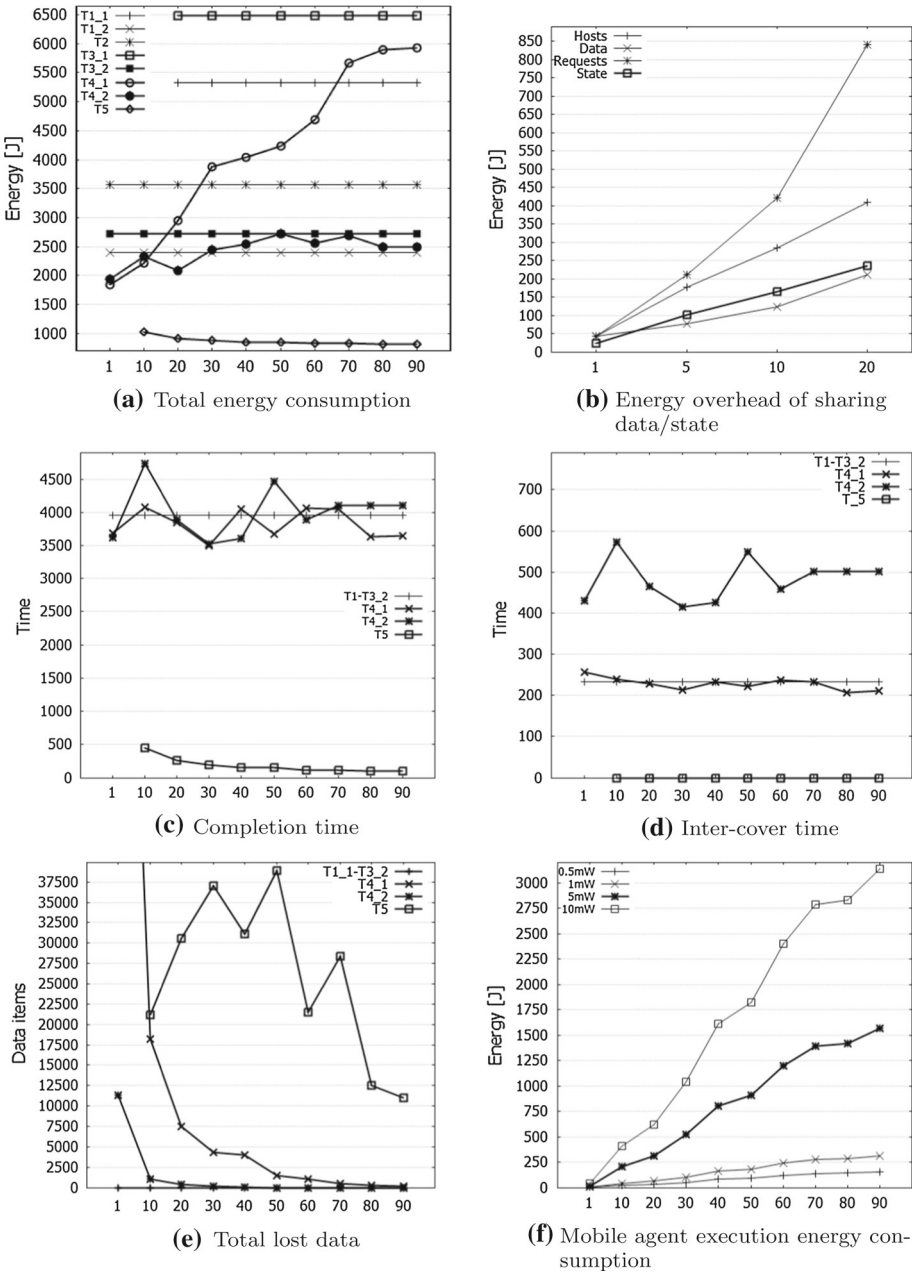


Fig. 4 Results of different parameters in the simulated campaigns

varies little when increasing the number of mobile agents (Fig. 5), where the application execution energy consumption dominates. Again, the total number of messages (Fig. 6) is reduced and agent migrations happen rarely due to the selected migration policy, based on the amount of unprocessed data in the devices. Task completion time remains about the same with a little variation (Fig. 4c). Inter-cover time increases significantly due to infrequent

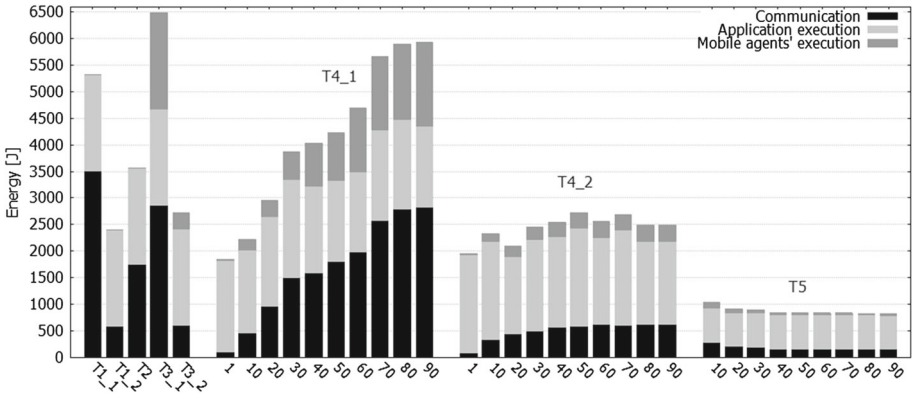


Fig. 5 Distribution of total energy consumption in each campaign

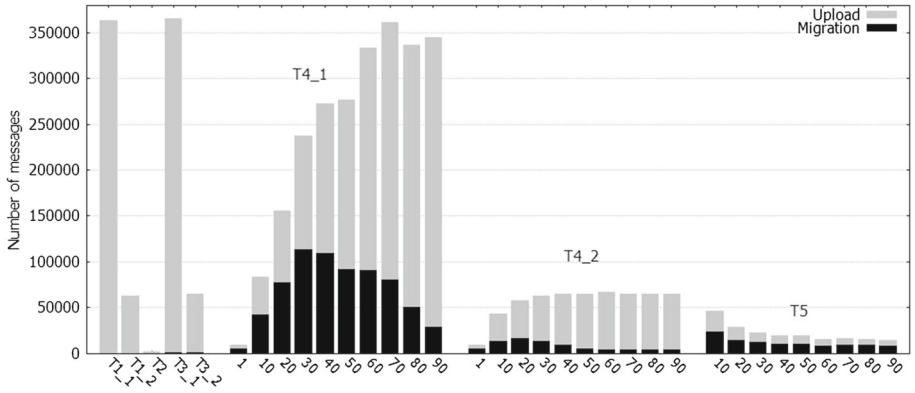


Fig. 6 Total number and distribution of different types of messages in each campaign

agent migrations, i.e. participants collect a large amount of data before it is processed. Lost data (Fig. 4e) approaches zero quickly as data is processed in large chunks in comparison to campaign T4_1.

Campaign T5 We observe that when increasing the number of participants, with a fixed number of mobile agents (Fig. 7), the campaign is completed faster as more participants cover the target area (Fig. 4c) thus the total energy consumption becomes low. As seen in Fig. 5, the application execution energy consumption overhead dominates. Inter-cover time becomes almost zero (Fig. 4d). The amount of lost data increases because participants collect data, but agents rarely migrate into the devices. Mobile agents process a large chunk of data in each migrated device, thus only a few messages are sent during the campaign (Fig. 6).

Campaign T6 This campaign operates as campaign T4_2, but we utilized ratio 4:1 in sharing agents states in MAS. This ratio was selected due to the number of smartphones in our real-world prototype in Sect. 6. Sharing the states of a small number of mobile agents introduces insignificant overhead energy consumption, as shown in Fig. 4b. If the campaigns T3_2 and T4_2 include the additional state sharing overhead of 20 agents (as in ratio 4:1), the total energy consumption is still lower than in the reference campaign T2.

Campaign T7 The number of data requests has a significant effect on the additional energy consumption of sharing, where the size of requested data chunk has less effect. But, when the number of agents and requests increases, we begin to observe a significant increase in energy

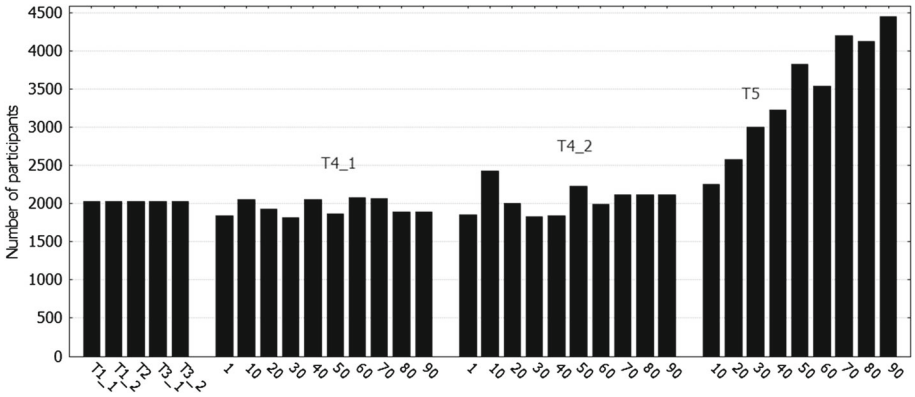


Fig. 7 Total number of participants in each simulated campaign

consumption. Even with real-time data sharing by 20 agents (as in ratio 4:1) in campaigns T3_2 and T4_2, the energy consumption is still lower than in the reference campaign T2.

5.3 Analysis of the simulation results

Generally, self-contained crowdsensing applications optimize data sampling, data processing and communications for the target application solely, thus can exhibit less total energy consumption (campaigns T1_1-T2). Nonetheless, modifying the campaign parameters is limited to the features of these applications. In comparison to the reference campaigns, the mobile agent based campaigns consume less energy with a small number of roaming mobile agents and with adaptive sampling. When the number of mobile agents increases, the energy consumption is about the same as in the best reference campaign. Thus, the reductions in energy consumption are related to the number of mobile agents. We observe the benefits of in-network data processing and agent migration, as the dominating factor in energy consumption becomes the crowdsensing application execution, as shown in Fig. 5. Campaign completion time varies slightly with the number of roaming mobile agents, remaining approximately the same in all campaigns except T5, due to its large number of participants. Results show that the amount of lost data stays on the same level with roaming mobile agents as with the reference campaigns, when the migration policy gives higher priority to devices with most unprocessed data. In multi-agent campaigns, state and data sharing introduces insignificant overhead in energy consumption with a small number of mobile agents, which increases the real-time utility of the data and task results. Nevertheless with mobile agents, we lose the ability to further process the raw data in the backend system in real-time. Still, raw data is stored in the participant device and can be later shared with the backend.

Some of the simulation parameters could be adjusted for further experiments, such as the message and data item sizes. Participant routes can be upgraded to resemble the real movement patterns of humans, with changing directions and pauses in movement. Campaign start and end criteria can be adjusted for different campaigns. The data transmission energy consumption in the simulations was divided between the value range (5–40 mJ/kB), as presented in [46], which is more realistic in real-world settings. Moreover, the energies used in application and mobile agent execution can be adjusted. We measured the effect of different agent execution energy overheads in the EE (from 0.5 to 10mW), as shown in Fig. 4f, where the approximated energy consumption was 5mW as reported in the real-world experiment.

This has a significant effect in the total energy consumption of the campaign, thus it becomes implementation optimization issue for the EE application. The mobile agent's task code can be presented in the composition in a scripting language, requiring interpretation overhead, or in the native language of the device. This has a significant impact on the agent execution energy consumption overhead.

Campaigns designed and implemented with mobile agent based in-network data processing transmit less data, which can reduce communication energy consumption but introduces data processing energy consumption overhead. Overall, the results suggest that with a relatively small number of mobile agents, the campaigns require less total energy than the reference campaigns, and otherwise do not introduce significant overhead energy. Mobile agents can share data and task results in real-time with a little overhead in energy consumption. Mobile agents-based campaigns should tolerate a small amount of lost data, which is an issue in all crowdsensing applications [31].

6 Prototype application

We implemented a software framework for mobile agent-based crowdsensing, based our previous work [34,35]. A small-scale real-world crowdsensing campaign for pedestrian flock detection, based on our previous work [2,33], was also designed and implemented. A Web server operates as the campaigner and a DRD runs as infrastructure service. For the pedestrian flock detection, the participants collect sensor data, i.e. conduct continuous Wi-Fi scan operation to detect different Wi-Fi access points' and their measured signal strengths. A cosine similarity measure is then applied to the data to cluster the participants into flocks during their movement. In related work, a data routing method in crowdsensing based on participant flocks was presented in [22].

The campaign is designed with two roles that are implemented as mobile agents. Mobile agent compositions are shown in Table 5. The sensing task is conducted in manual mode where sensing is continuous, ie. the 'wifi_vector" mobile agent processes available new data when it migrates into the device. In its task, it calculates the averages of measured signal strengths' of each Wi-Fi access point for a time window that are exposed as data vector in its state. EEs in the devices store the task results into databases, therefore, the most current result can be returned for queries, even if the agent has migrated away from the device. The "similarity" mobile agent retrieves the data vectors by one-shot queries from the other agents and then calculates the cosine similarity measure between each data vector. That result is then uploaded by the task to the campaigner after each iteration. The campaigner then determines and presents the detected flocks in the Web service. The purpose of this design is to reduce the energy consumption of the participating devices (the global measure) and to demonstrate MAS operation by sharing task results in the campaign in runtime, thus the device hosting "similarity" agent is not part of the data collection.

See Fig. 8 for illustration of the application. The campaign operates as follows. Each participant registers its resources into the DRD (1). The campaigner performs lookup to the DRD for resource "wifi" (the name for the Wi-Fi scan operation as sensor), until a sufficient number of participants has been found (2). The campaign requires five participants, as explained in the next section, and no other recruitment method was utilized. As the agent task code has already been written by the campaigner, it is only needed to add the located available resources (i.e. participants' devices) into the mobile agent composition and inject the agent into the system. The campaigner maintains the designed ratio between "similarity"

Table 5 The two mobile agents utilized in the prototype application

Agent	Similarity	Wifi_vector
Code	"Process"	"Filter"
	similarity[0] = cos_sim(p2, p3)	def averages(wifi_data):
	similarity[1] = cos_sim(p2, p4)	…
	similarity[2] = cos_sim(p2, p5)	"Process"
	similarity[3] = cos_sim(p3, p4)	wifi_vector.add(averages(wifi))
	similarity[4] = cos_sim(p3, p5)	
	similarity[5] = cos_sim(p4, p5)	
Resource	"Local"	"Local"
	"Remote"	wifi = phone_2/wifi
	p2 = phone_2/wifi_vector	wifi = phone_3/wifi
	p3 = phone_3/wifi_vector	wifi = phone_4/wifi
	p4 = phone_4/wifi_vector	wifi = phone_5/wifi
	p5 = phone_5/wifi_vector	"Remote"
	"Static"	"Static"
State	Similarity = […]	wifi_vector = […]
Metadata	[…]	[…]

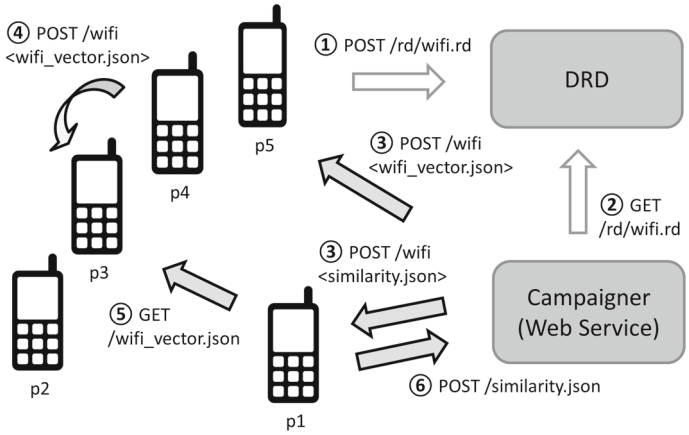


Fig. 8 Mobile agent-based crowdsensing application for pedestrian flock detection

and "wifi_vector" agents by injecting mobile agents into the system (3). Then "wifi_vector" mobile agent migrates according to its fixed itinerary (4) and "similarity" agent operates as resident agent, retrieving the "wifi_vector" results from the smartphones (5) and uploading its results to the campaigner (6). The campaign end criterion is defined as a running period of 15 minutes. When the time has elapsed, the campaigner deletes its mobile agents by performing lookup into the DRD to locate the mobile agents and sends delete messages to the hosts.

We have implemented an EE application for Android operating system in Java in our previous work [34] that was extended here to facilitate MAS. The EE utilizes HTTP for communication atop Wi-Fi with the presented agent interaction protocols. The data format

is JSON. The agent task codes where implemented in Python although the scripting service implementation also supports JavaScript and a number of other scripting languages[3]. When the EE application is started, it registers the device and its resources, i.e. sensors, to the DRD. The registration message format [35] enables an unlimited set of application-specific parameters in the resource description, all of which can be queried separately. When device resources change, e.g. after mobile agent migration, EE updates the resource description in the DRD. The EE user interface visualizes the remaining smartphone battery and current power consumption of the utilized sensors. Participation can be canceled from the user interface anytime.

For each mobile agent, a thread is started for its execution, where scheduling and execution of multiple threads are done by the operating system. The data in the device can be accessed by all agents simultaneously, but there are no conflict resolution mechanisms for the actions performed by the agents. Once a task execution has started, it can't be stopped externally but only from the task code. When a mobile agent migrates into a device, the EE operates as follows. First, it retrieves the requested resources as described in the agent composition. Local data is retrieved from the database in the device and external data from the remote host. Task execution is halted while it waits for data. Then, a runnable script code is generated from the task code and sent to the scripting service for execution. If the code includes local or external actions, these are communicated by the scripting engine to the EE as messages describing the action. This way, the task can send messages, dispatch events or operate external components. The implementation does not support retrieving external data during the script execution, all data must be requested through the resource segment before execution. After the task execution, control is returned to the EE that updates the agent state in the composition and stores results into the database with a timestamp, enabling to query the results of a particular participant regardless of where mobile agents are currently hosted. This way, we mitigate the agent tracing problem. At this point, the EE updates the mobile agent's resource description in the DRD to facilitate exposing the new state. Now, the mobile agent is ready to migrate to the new host or execute the next iteration in this device, according to the migration policy.

6.1 Evaluation of the prototype application

We run the pedestrian flock detection with two different campaign designs and parameters. Both designs had one "similarity" mobile agent that migrated into a device at the campaign start. The first design had one "wifi_vector" mobile agent, that migrated between the other devices and processed their collected data of designed period as given below. The second design had four "wifi_vector" agents, that migrated into the other devices at campaign start and operated as resident agents. We assumed the ratio of 1:4 between "similarity" and "wifi_vector" agents due to the number of available smartphones for our experiments. For evaluation, we added one reference campaign. Thus the evaluated campaigns were: (1) "Upload" is a reference campaign with no agents where the smartphones upload raw data to campaigner periodically, as in the simulated campaign T1_1; (2) "MA resident", where one "similarity" agent retrieves data from four resident "wifi_vector" agents, as in the simulated campaign T3_1; and (3) "MA migrate", where one "similarity" agent retrieves data from one migrating "wifi_vector" agent as in the simulated campaign T4_1. Moreover, we experimented with different agent migration intervals by changing this parameter in campaigns to demonstrate the effect on the total campaign energy consumption: (1) every scan result; (2) 20 s; (3) 1 min; (4) 4 min; and (5) 15 min. In other words, the agent migrates into the devices

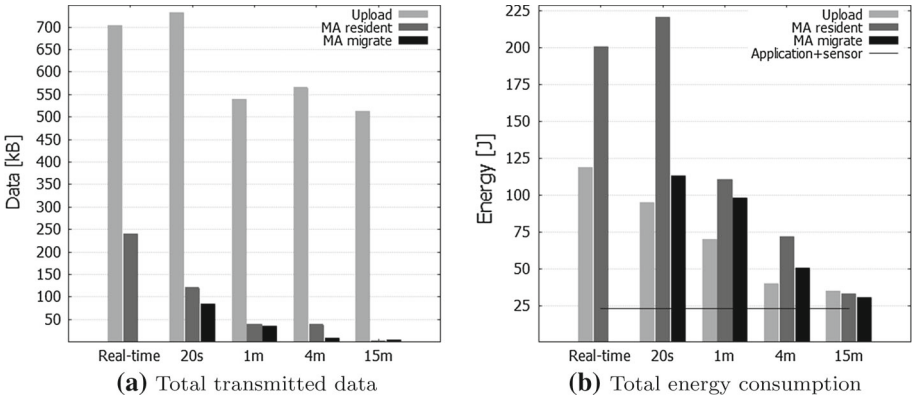
[3] https://github.com/damonkohler/sl4a

in this interval and the task algorithm processes data for this period. Campaign "Upload" with the interval of 15 min corresponds to simulated campaign T2 with additional data processing.

We utilized five Android smartphones as the devices: (1) two Samsung Galaxy S3 (GT I-9300) with Android 4.3, one for each agent type, (2) two Samsung Galaxy S4 Mini (GT-I9195) with Android 4.2.2 and Android 4.4.2 executing the "wifi_vector" agents, and (3) one Samsung Galaxy S5 Mini (SM-G800F) with Android 4.4.2 executing the "wifi_vector" agent. The energy comsumption monitoring of the EE application was done with Powertutor[4] application, although it is implemented for earlier version of Android operating system and hardware, thus inaccuracies are possible. From each device, we collected energy consumption data and amount of transmitted data in each campaign. Moreover, the Wi-Fi radio energy consumption monitoring by Powertutor was not directly shown for the selected smartphone models. The energy consumption data in the participant devices is not available in runtime to the EE. We noticed significant differences in the energy consumption and mobile agent execution energy consumption overhead in the EE application in different smartphones. The energy consumption of different smartphone hardware models and components changes between generations, as also discussed in [46]. Moreover, the behavior of the application and varying network conditions in the public Wi-Fi network have an effect. For the measured energy consumption data, the Galaxy S3 smartphones presented the maximum standard deviation of 7J, the Galaxy S5 Mini 8J and the Galaxy S4 Mini up to 29J. Android 4.2.2. operating system appeared slightly more stable in this respect.

Figure 9a indicates that this campaign design, facilitating mobile agent based in-network data processing, significantly reduced the amount of transmitted data in comparison to the reference "Upload" campaign. This contributes towards less communication energy consumption and less data processing in participating devices. The energy consumption of evaluated campaigns is shown in Fig. 9b that shows for each campaign the sum of average energy consumption of each participant (i.e. smartphone model). The "Application+sensor" value shows the measured application execution energy consumption that was also utilized in the simulated campaigns. The lowest energy consumption is with the campaigns with a 15 min data upload period, where data is once processed and uploaded at the end of the campaign. The campaign "MA migrate" (relates to campaign T4_1) consumes less energy than the campaign "MA resident" (as campaign T3_1). We observe that the amount of transmitted data is considerably lower in campaign "MA migrate", but it consumes slight more energy in comparison to the reference campaign "Upload". The reason for this is the extra energy consumed in agent execution, which we can't confirm as we cannot reliably separate the energy consumption of the different components of the EE application with Powertutor and heterogeneous hardware. This EE application design and implementation issue becomes tradeoff between real-time utility of data and agent execution energy overhead, as it may eliminate the benefit of communication energy reduction. Nevertheless, it can be assumed that implementing the agent task code in native language of the platform would considerably reduce the agent execution energy consumption. To address this issue in the simulations, we considered different agent execution energy consumptions (Fig. 4f). As shown in the figure, the effect of the agent execution to the total campaign energy consumption can be significant. When the agent execution overhead becomes large, a solution could be to share the data and task results instead of local data processing in each device. With mobile agents, the data and results are shared in runtime at the devices or the task is relocated by agent migration, in comparison to related work in [54], that processes data locally only once and shares results between applications only at the backend.

[4] http://ziyang.eecs.umich.edu/projects/powertutor/



(a) Total transmitted data **(b)** Total energy consumption

Fig. 9 Evaluation results of the real-world campaigns

Smartphones today include several communication interfaces, such as 3G, Wi-Fi and Bluetooth, with their individual energy consumption profiles. For each interface, energy is consumed in keeping the radio interface on, scanning and setting up connections. Wi-Fi was determined more energy efficient than 3G in [22]. Opportunistically uploading data in bursts and otherwise keeping the radio interface sleeping could signifigantly decrease the energy consumption [22,46,51]. The issue with Wi-Fi is that the access points can be sparsely deployed in urban environments. With Bluetooth, the drawbacks are frequent connection set-ups and a limited number of simultaneous communication links.

Lower layer communication protocol optimizations, such as optimized payload size or send buffer size, are possible, but the evaluation metrics would be different from layer to layer and are outside the focus of this paper. Decreasing data sending intervals requires more energy, but on the other hand, the energy consumption per bit is smaller [51]. Sending data consumes considerable more power than receiving data [51]. Changes in the voltage level of the battery also affect the power consumption [51]. Optimizing data transmission strategy in smartphones differs between hardware, operating systems and device contexts [46]. With regard to Android-based smartphones, the operating system, other processes and applications try to communicate with their servers in regular basis, which may additionally distort the energy consumption results [46]. Energy consumption profiles of the smartphone sensors are different for each sensor and may have significant effect on the energy consumption, which is true for all crowdsensing applications. Concrete generalized conclusions are therefore difficult to draw. In this evaluation, we did not consider varying network conditions nor communication latencies. Participant device failures, such as depleted battery, are not addressed in this prototype.

In conclusion, despite of the considerable agent execution energy consumption, we observe similar results in the real-world implementation than in the simulations. Less data is transmitted and less messaging needed with in-network processing. Reductions in energy consumption are seen with different campaign designs of roaming mobile agents, in comparison with the reference campaign of uploading all data in real-time to the campaigner. Alike, the real-world campaign utilized only a few mobile agents. This suggests that mobile agent-based campaigns are the most beneficial for opportunistic data collection style, where agents migrate into devices infrequently. This sacrifices real-time data utility for the reduced agent execution overhead, which can be addressed in EE application implementation. When the overhead is lower, there is more room (energy-wise) for more mobile agents to operate.

With different global measure and agent migration policy, the campaigner could emphasize real-time data utility or minimize participants' discomforts, for example. When the agent execution overhead is smaller, campaign execution could be significantly more effective in both reducing energy consumption and increasing real-time utility. More complex campaigns that make better use of MAS features could be implemented, presumably with additional energy consumption for agent operations. Referring back to the earlier discussion, reducing the energy consumption of the crowdsensing campaigns alone justifies the use of mobile agents as the participants' devices are generally battery-operated smartphones, provided that comparable results for the campaigns are produced.

7 Related work

Related work in mobile crowdsensing and participatory sensing are extensive and methods are too numerous to conclusively discuss here. We consider the relevant work in crowdsensing frameworks, including software agent-based solutions.

The ParticiPact mobile crowdsensing platform [7] enables large-scale campaigns with fine-grained sensing actions, where the sensors are duty-cycled and data sharing energy overhead minimized. Actions are completed within the given timeframe in the target location, with the most limited subset of available participants that are selected based on their profiles. Sensing tasks are deployed as OSGi bundles to both client- and server-side, but the system does not facilitate migration. The client-side manages task and sensing actions for the participants and provides long-term data storage in local databases, but copies are sent to backend. The backend system manages all tasks, performs large-scale data post-processing and data mining. For each action, incentives are available for the participants.

Context-aware mobile crowdsensing service-oriented architecture with social networking is introduced in [23] that integrates multidimensional data sources, such as personal, environmental and social, with crowdsensing applications and Web services to provide higher-level contextual data with a common ontology. Context is utilized to improve the usability of the services in the system. A cloud platform deploys and coordinates campaigns, and exposes results in a Web service. Campaigns are executed with software agents in the participating devices. This work is based on TripleS [24] that utilizes a cloud platform to coordinate crowdsensing tasks for social networking services. A service-oriented architecture framework in the mobile devices realizes a set of services, implemented as application-specific resident agents: local sensing service, crowdsensing service, context-awareness service, interfaces to the cloud platform and for social networking, and management of services. Mobile agents are utilized for augmenting resident agents and transferring their accumulated results.

In [56], energy-efficient participatory sensing task allocation framework is presented. The framework facilitates a centralized task allocation and scheduling component, that handles both offline and online data queries. The idea is to allocate temporally overlapping queries of the same data to the same participating devices. Device allocation is based first on the devices which can cover the task entirely, secondly on devices with smallest aggregated sensing time, and thirdly on devices with least increased sensing time. Tasks are then executed in devices in temporal order based on task start time.

In [21], a service-oriented middleware for participatory sensing is presented. Devices periodically advertise their resources and future paths to a system registry. When the registry receives queries for data, it selects the relevant devices and composes a data acquisition

service from them. The devices are considered for participation only if they increase the sensing coverage, i.e. their expected moving path is not redundant or an already registered device or substitute composition does not exist for the path. The middleware aims at minimum sensing coverage to limit the number of participating devices and tries to reduce total energy consumption. User queries are compiled with a common ontology.

Zhao et al. [55] utilize the trajectory histories of selected devices in participatory sensing to provide energy-efficient coverage. Trajectories assist in offline participant selection, and in online, to adapt the sampling frequency with spatiotemporal correlations among sensing data. Each device knows its location and disseminates its location to all other devices with epidemic data exchange. Their results, collected with real-world human and taxi mobility traces, imply that human mobility traces are stable enough for meaningful predictions of future traces.

Matador [9] is a crowdsensing framework that embeds context-awareness to the sensing task execution. Campaigners describe tasks through a Web application as XML files that give the required context and describe the action for data collection or for users. The smartphone application contains a task list, that is periodically synchronized with server-side. User context, such as location, is adaptively sampled in smartphones with modifiable accuracy and rate. The server-side creates the visualizations of the task results.

Layered virtual machine-based crowdsensing application deployment model was presented in [52]. At the lowest layer, mobile devices and crowdsensing applications forward their data to "proxy" virtual machines in the second layer. The second layer comprises distributed cloud infrastructure, where the proxy virtual machines migrate according to the mobility of the user. Other virtual machines then perform application-specific data processing and forward the results to the top layer. Coordinating entities in the top layer, i.e. application servers in the cloud infrastructure, manage application execution. The infrastructure includes a global registry to discover proxy virtual machines for tasks.

A geo-social crowdsensing platform was presented in [8]. The idea is to build time-variant resource maps that enable dimensioning people's involvement and sensing accuracy in the data backend to assist in designing crowdsensing campaigns. The maps are constructed by analyzing the results of mobile sensing actions and the contexts of potential participants, after which, tasks are assigned automatically for the participants.

In [42], is presented a Web-based framework for users to deliver simple sensing tasks to their smartphones. A set of programming abstractions are provided for tasks and for composing higher level abstractions from the tasks. The tasks, stored in a system repository, are precompiled in servers and heuristically partitioned to execution in either on the smartphone or in remote servers if the task requires input from other system devices.

Medusa [41] is a programming framework for crowdsensing applications with a macroprogramming language based on the master-worker pattern. Applications are described as stages, concurrently executed in the smartphones and a cloud platform. The results of each stage are inputs to the next stages. The master is only informed when the task is complete. The stages are reusable from a repository.

MECA [54] is a framework, where the idea is to share the results of data processing between applications in a way that each primitive operation for the data is done only once in the participating devices. A layered architecture is utilized, where the "phenomena" layer receives high-level data collection specifications from applications, selects appropriate nodes on the "edge" layer at the network edge that can provide data for the application and then returns the data to the applications. "Edge" layer then selects and coordinates a subset of local devices in the "data" layer, that have resources to provide the particular data. Each device runs a software agent to execute the instructions for data collection and processing.

Prism [15] addresses running precompiled binary code in a set of smartphones as tasks. The phones are centrally orchestrated to co-operatively participate in mobile sensing applications, but without co-operation. The smartphones register their current location and available resources into a server. The server then determines suitable participants for the task, based on the given set of predicates, then pushing the tasks to the participants.

An agent-based system for crowdsensing was presented in [48]. Raw sensor data is provided by user agents in the phones, at the same time, monitoring the user performance. Then, data is processed by local system agents by instructions from a coordination agent. Verification agents validate the processed data and the user performance, providing feedback for participant selection. Application-specific event monitoring agents send notifications to the coordination agent and start services for sensing tasks based on detected events. A central coordination agent stores the data into a database, communicates with all agents and system operators to control the tasks and provides access end-point for system operators.

Darwin [37] is a framework for collaborative mobile phone sensing with local data analytics on the phone and further analysis in the backend. The data processing algorithms are shared between participating phones to globally enhance the algorithms on each phone to increase sensing accuracy.

In [53], a mobile crowdsourcing platform was presented, which allows users to post their own tasks to the system. The tasks exploit available mobile workers in the task execution. Data queries can be made through a mobile application to existing crowdsourcing tasks in the system. The uploaded task results are then made available in the mobile application.

A privacy preserving software framework for crowdsensing was presented in [14] that protects the privacy of participants and guarantees data integrity. A mixing channel is used to anonymize data sources in received data during upload. Framework also features a registration authority that ensures authenticity of the system components. Tasks are described in their specific high-level language.

8 Discussion and conclusion

In this work, we introduce mobile agents-based MAS for crowdsensing. A prominent benefit is that mobile agents provide decentralized and autonomous campaign execution that enables mobile agents to consider resource availability and participant related issues in their operations. Mobile agents distribute computation and communication load into the system based on their tasks and control algorithms. Moreover, mobile agents can negotiate complex global measures that guide the campaign execution, where examples include minimizing total energy consumption, minimizing participants' discomforts or maximizing data utility in real-time.

We described a software framework to realize mobile agent-based crowdsensing campaigns, supporting both participatory and opportunistic approaches. The framework adopts REST principles and resource-oriented architecture that enables the seamless integration of campaigns into the Internet. HTTP is employed as a universal communication protocol in the framework, including agent interaction protocols. The framework facilitates the online model, where multiple campaigns execute asynchronously relying their context based resource availability. Campaigns are designed as a MAS with different targets, roles and interactions. Role-based interactions promote the reuse of data and agents' tasks results through real-time sharing. Roles embrace their specific data collection and analysis tasks that are realized as mobile agents. Mobile agents execute, control and monitor the campaign through

their role-based tasks autonomously in the opportunistic network of participants' smartphones. Mobile agents' tasks include negotiating participants' personal boundaries with the campaign requirements, taking into account the individual context and data privacy management. Participants' behaviors are monitored by the agents and appropriate compensation calculated for their resource usage and discomforts.

We conducted a set of simulations to study the characteristics of mobile agent-based campaigns in comparison to the "traditional" crowdsensing approaches. The simulation results indicate that campaigns designed with a relatively small number of mobile agents performs best in terms energy consumption. The amount of data transmitted and messaging in the campaign reduces considerably with mobile agent-based in-network data analysis and with campaign-specific adaptive sampling algorithms. The sharing of the task results in real-time in the campaign introduces insignificant overhead to the campaign execution at the same time increasing real-time data utility. However, the participants' mobility patterns and agent migration policies have impact on this aspect. These results enable to further consider MAS in crowdsensing, as several issues could be studied with simulations: sophisticated campaign control, negotiation capabilities of mobile agents, different task assignment policies [8], event-driven tasking, data quality [45], participant reputation in recruitment and robust campaign execution.

We implemented a small-scale real-world crowdsensing campaign based on mobile agents atop the implementation of the framework. The target application was pedestrian flock detection, designed with two roles realized with mobile agents interacting with each other. The real-world evaluation results are similar to the simulation results. However, we experienced relatively large agent execution energy consumption overhead in the EE application, due to its implementation. This overhead could be reduced with reconsideration of the EE implementation. When the overhead is smaller, mobile agent based campaign execution could be significantly more effective. Notwithstanding, the optimization of application and communication energy consumption in smartphones is a complex task. Hardware design with static and dynamic characteristics of the utilized components have an effect on this. This information is crucial for the optimized campaign design, therefore, energy consumption information in participants' devices should be exposed.

Referring back to the challenges listed in Table 1, the implemented framework features unified system architecture and standardized communication protocol. Easy campaign deployment is facilitated through seamless connection to the Internet and campaign monitoring in real-time is featured through real-time upload of results. MAS based campaign design facilitates cooperation and sharing of data in real-time. Data quality issues can be addressed either in the device or by the campaigner. Concerning participant-related issues, real-time resource availability is exposed in the framework, but no other means to recruit, evaluate or compensate participants are implemented. Privacy concerns are addressed by exposing only a selected set of data features into the system, but participants can not negotiate their individual boundaries. The implemented mobile agents did not utilize event detection in data collection and to react to changes in the system or smartphone use context. The real-world system did not feature multiple simultaneous campaigns, nor integration to social networking systems. The framework addresses guidelines for mobile crowdsensing systems [8]: minimal delay in producing the stream of information, minimal computing overhead with fast feedback mechanism, data transmission in phases, and complete data management cycle.

This work suggests that mobile agent-based campaigns are most beneficial for opportunistic crowdsensing data collection style. Benefits are evident to campaigns that are designed with infrequent agent migration and utilize data sharing. Mobile agent migration and in-network data analysis significantly reduces the amount of transmitted data and contributes

to reduced energy consumption in campaigns. Reduced energy consumption alone justifies the use of mobile agents, provided that comparable results for the campaigns are produced. As a future work, more complex campaigns that make better use of MAS features are to be studied. Sophisticated campaign control with more complex agent interaction protocols and machine learning algorithms play a role in the autonomous crowdsensing campaign execution as MAS. We believe that this approach can potentially become one of the integral data acquisition methods in pervasive computing environments.

References

1. Abdelzaher, T., Anokwa, Y., Boda, P., Burke, J., Estrin, D., Guibas, L., et al. (2007). Mobiscopes for human spaces. *IEEE Pervasive Computing*, *6*, 20–29.
2. Álvarez Lacasia, J., Leppänen, T., Iwai, M., Kobayashi, H., & Sezaki, K. (2013). A method for grouping smartphone users based on wi-fi signal strength. In: *IPSJ Forum on Information Technology* (Vol. J-032).
3. Bellifemine, F., Poggi, A., & Rimassa, G. (2001). Developing multi-agent systems with a FIPA-compliant agent framework. *Software—Practice and Experience*, *31*(2), 103–128.
4. Burke, J, Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S., et al. (2006). Participatory sensing. In *4th ACM conference on embedded networked sensor systems, 1st workshop on world-sensor-web: Mobile device centric sensory networks and applications*. New York: ACM.
5. Campbell, A., Eisenman, S., Lane, N., Miluzzo, E., Peterson, R., Lu, H., et al. (2008). The rise of people-centric sensing. *IEEE Internet Computing*, *12*(4), 12–21.
6. Campbell, A., & Wu, A. (2011). Multi-agent role allocation: issues, approaches, and multiple perspectives. *Autonomous Agents and Multi-agent Systems*, *22*(2), 317–355.
7. Cardone, G., Cirri, A., Corradi, A., & Foschini, L. (2014). The participact mobile crowd sensing living lab: The testbed for smart cities. *IEEE Communications Magazine*, *52*(10), 78–85.
8. Cardone, G., Foschini, L., Bellavista, P., Corradi, A., Borcea, C., Talasila, M., et al. (2013). Fostering participaction in smart cities: A geo-social crowdsensing platform. *IEEE Communications Magazine*, *51*(6), 112–119.
9. Carreras, I., Miorandi, D., Tamilin, A., Ssebaggala, E., & Conci, N. (2013). Matador: Mobile task detector for context-aware crowd-sensing campaigns. In *IEEE international conference on pervasive computing and communications workshops (PerCom2013 workshops)* (pp. 212–217).
10. Chen, M., Kwon, T., Yuan, Y., & Leung, V. (2006). Mobile agent based wireless sensor networks. *Journal of Computers*, *1*(1), 14–21.
11. Chon, Y., Lane, N., Kim, Y., Zhao, F., & Cha, H. (2013). Understanding the coverage and scalability of place-centric crowdsensing. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing—UbiComp '13* (pp. 3–12).
12. Christin, D., Reinhardt, A., Kanhere, S., & Hollick, M. (2011). A survey on privacy in mobile participatory sensing applications. *Journal of Systems and Software*, *84*, 1928–1946.
13. Conti, M., & Giordano, S. (2014). Mobile ad hoc networking: Milestones, challenges, and new research directions. *IEEE Communications Magazine*, *52*, 85–96.
14. Cornelius, C., Kapadia, A., Kotz, D., Peebles, D., Shin, M., & Triandopoulos, N. (2008). Anonysense: Privacy-aware people-centric sensing. In *Proceedings of the 6th international conference on Mobile systems, applications, and services, MobiSys '08* (pp. 211–224).
15. Das, T., Mohan, P., Padmanabhan, V., Ramjee, R., & Sharma, A. (2010). Prism: Platform for remote sensing using smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services, MobiSys '10* (pp. 63–76).
16. Dikaiakos, M., Kyriakou, M., & Samaras, G. (2001). Performance evaluation of mobile-agent middleware: A hierarchical approach. In G. Picco (Ed.), Mobile agents, lecture notes in computer science (Vol. 2240, pp. 244–259). Atlanta: Springer.
17. Eberle, J., Yan, Z., & Aberer, K. (2013). Energy-efficient opportunistic collaborative sensing. In *10th IEEE international conference on mobile ad-hoc and sensor systems* (pp. 374–378).
18. Estrin, D. (2010). Participatory sensing: Applications and architecture. *IEEE Internet Computing*, *14*, 12–14.
19. Ganti, R., Ye, F., & Lei, H. (2011). Mobile crowdsensing: Current state and future challenges. *IEEE Communications Magazine*, *49*(11), 32–39.

20. Guo, B., Yu, Z., Zhou, X., & Zhang, D. (2014). From participatory sensing to mobile crowd sensing. In *IEEE international conference on pervasive computing and communications workshops (PerCom Workshops)* (pp. 593–598).
21. Hachem, S., Pathak, A., & Issarny, V. (2014). Service-oriented middleware for large-scale mobile participatory sensing. *Pervasive and Mobile Computing*, *10*, 66–82.
22. Higuchi, T., Yamaguchi, H., Higashino, T., & Takai, M. (2014). A neighbor collaboration mechanism for mobile crowd sensing in opportunistic networks. In *IEEE international conference on communications (ICC2014)* (pp. 42–47).
23. Hu, X., Li, X., Ngai, E., Leung, V., & Kruchten, P. (2014). Multidimensional context-aware social network architecture for mobile crowdsensing. *IEEE Communications Magazine*, *52*(6), 78–87.
24. Hu, X., Liu, Q., Zhu, C., Leung, V., Chu, T., & Chan, H. (2013). A mobile crowdsensing system enhanced by cloud-based social networking services. In *Proceedings of the first international workshop on middleware for cloud-enabled sensing, MCS '13* (pp. 1–6). New York: ACM.
25. Jaimes, L., Vergara-Laurens, I., & Labrador, M. (2012). A location-based incentive mechanism for participatory sensing systems with budget constraints. In *IEEE international conference on pervasive computing and communications* (pp. 103–108).
26. Kanjo, E., Bacon, J., Roberts, D., & Landshoff, P. (2009). Mobsens: Making smart phones smarter. *IEEE Pervasive Computing*, *8*(4), 50–57.
27. Khan, W., Xiang, Y., Aalsalem, M., & Arshad, Q. (2013). Mobile phone sensing systems: A survey. *IEEE Communications Surveys & Tutorials*, *15*(1), 402–427.
28. Koutsopoulos, I. (2013). Optimal incentive-driven design of participatory sensing systems. In *32nd IEEE international conference on computer communications* (pp. 1402–1410).
29. Lane, N. (2012). Community-aware smartphone sensing systems. *IEEE Internet Computing*, *16*(3), 60–64.
30. Lane, N., Eisenman, S., Musolesi, M., Miluzzo, E., & Campbell, A. (2008). Urban sensing systems: Opportunistic or participatory? In *Proceedings of the 9th workshop on Mobile computing systems and applications* (pp. 11–16). New York: ACM.
31. Lane, N., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., & Campbell, A. (2010). A survey of mobile phone sensing. *IEEE Communications Magazine*, *48*(9), 140–150.
32. Lange, D., & Oshima, M. (1999). Seven good reasons for mobile agents. *Communications of the ACM*, *42*, 88–89.
33. Leppänen, T., Álvarez Lacasia, J., Ramalingam, A., Liu, M., Harjula, E., Närhi, P., et al. (2013). Interoperable mobile agents in heterogeneous wireless sensor networks. In *Proceedings of the 11th ACM conference on embedded networked sensor systems (SenSys'13)*. New York: ACM.
34. Leppänen, T., Liu, M., Harjula, E., Ramalingam, A., Ylioja, J., Närhi, P., et al. (2013). Mobile agents for integration of internet of things and wireless sensor networks. In *IEEE international conference on systems, man, and cybernetics (SMC)* (pp. 14–21).
35. Liu, M., Leppänen, T., Harjula, E., Ou, Z., Ramalingam, A., Ylianttila, M., et al. (2013). Distributed resource directory architecture in machine-to-machine communications. In *IEEE 9th international conference on wireless and Mobile computing, networking and communications (WiMob)* (pp. 319–324).
36. Ma, H., Zhao, D., & Yuan, P. (2014). Opportunities in mobile crowd sensing. *IEEE Communications Magazine*, *52*(8), 29–35.
37. Miluzzo, E., Cornelius, C., Ramaswamy, A., Choudhury, T., Liu, Z., & Campbell, A. (2010). Darwin phones: The evolution of sensing and inference on mobile phones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services, MobiSys '10* (pp. 5–20). New York: ACM.
38. Miluzzo, E., Lane, N., Fodor, K., Peterson, R., Lu, H., Musolesi, M., et al. (2008). Sensing meets mobile social networks: The design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM conference on embedded network sensor systems, SenSys '08* (pp. 337–350). New York: ACM.
39. Niwa, J., Okada, K., Okuda, T., & Yamaguchi, S. (2013). Mpsdatastore: A sensor data repository system for mobile participatory sensing. In *Proceedings of the second ACM SIGCOMM workshop on Mobile Cloud Computing, MCC '13* (pp. 3–8). New York: ACM.
40. Pournajaf, L., Xiong, L., & Sunderam, V. (2014). Dynamic data driven crowd sensing task assignment. *Procedia Computer Science*, *29*, 1314–1323.
41. Ra, M. R., Liu, B., La Porta, T., & Govindan, R. (2012). Medusa: A programming framework for crowd-sensing applications. In *Proceedings of the 10th international conference on Mobile systems, applications, and services, MobiSys '12* (pp. 337–350). New York: ACM.
42. Ravindranath, L., Thiagarajan, A., Balakrishnan, H., & Madden, S. (2012). Code in the air: Simplifying sensing and coordination tasks on smartphones. In *Proceedings of the twelfth workshop on Mobile computing systems & applications, HotMobile '12*. New York: ACM.

43. Reddy, S., Estrin, D., & Srivastava, M. (2010). Recruitment framework for participatory sensing data collections. In: P. Floren, A. Krger, & M. Spasojevic (Eds.), *Pervasive Computing*. Lecture Notes in Computer Science (Vol. 6030, pp. 138–155). Berlin: Springer.
44. Reddy, S., Samanta, V., Burke, J., Estrin, D., Hansen, M., & Srivastava, M. (2009). Mobisense—Mobile network services for coordinated participatory sensing. In *IEEE International symposium on autonomous decentralized systems, ISADS '09* (pp. 1–6).
45. Riahi, M., Papaioannou, T., Trummer, I., & Aberer, K. (2013). Utility-driven data acquisition in participatory sensing. In *Proceedings of the 16th international conference on extending database technology, EDBT '13* (pp. 251–262).
46. Rice, A., & Hay, S. (2010). Measuring mobile phone energy consumption for 802.11 wireless networking. *Pervasive and Mobile Computing*, *6*(6), 593–606.
47. Shilton, K. (2009). Four billion little brothers? Privacy, mobile phones, and ubiquitous data collection. *Communications of the ACM*, *7*, 40–47.
48. Sun, Y., & Nakata, K. (2010). An agent-based architecture for participatory sensing platform. In *IEEE 4th international universal communication symposium (IUCS)* (pp. 392–400).
49. Tsujimori, T., Thepvilojanapong, N., Ohta, Y., Zhao, Y., & Tobe, Y. (2014). History-based incentive for crowd sensing. In *Proceedings of the international workshop on web intelligence and smart sensing, IWWISS '14* (pp. 1–6). New York: ACM.
50. Tuncay, G., Benincasa, G., & Helmy, A. (2013). Participant recruitment and data collection framework for opportunistic sensing: A comparative analysis. In *Proceedings of the 8th ACM MobiCom workshop on challenged networks, CHANTS '13* (pp. 25–30). New York: ACM.
51. Wang, L., & Manner, J. (2010). Energy consumption analysis of wlan, 2g and 3g interfaces. In *Proceedings of the 2010 IEEE/ACM Int'L conference on green computing and communications & Int'L conference on cyber, physical and social computing, GREENCOM-CPSCOM '10* (pp. 300–307). IEEE Computer Society.
52. Xiao, Y., Simoens, P., Pillai, P., Ha, K., & Satyanarayanan, M. (2013). Lowering the barriers to large-scale mobile crowdsensing. In *Proceedings of the 14th workshop on Mobile computing systems and applications, HotMobile '13* (pp. 9:1–9:6). New York: ACM.
53. Yan, T., Marzilli, M., Holmes, R., Ganesan, D., & Corner, M. (2009). mcrowd: A platform for mobile crowdsourcing. In *Proceedings of the 7th ACM conference on embedded networked sensor systems* (pp. 347–348). New York: ACM.
54. Ye, F., Ganti, R., Dimaghani, R., Grueneberg, K., & Calo, S. (2012). Meca: Mobile edge capture and analysis middleware for social sensing applications. In *Proceedings of the 21st international conference companion on World Wide Web, WWW '12 companion* (pp. 699–702). New York: ACM.
55. Zhao, D., Ma, H., & Liu, L. (2014). Energy-efficient opportunistic coverage for people-centric urban sensing. *Wireless Networks*, *20*(6), 1461–1476.
56. Zhao, Q., Zhu, Y., Zhu, H., Cao, J., Xue, G., & Li, B. (2014). Fair energy-efficient sensing task allocation in participatory sensing with smartphones. In *INFOCOM 2014 proceedings IEEE* (pp. 1366–1374).